DOCUMENT RESUME

ED 388 307 IR 017 456

AUTHOR Zhou, Gang
TITLE Curriculum Knowledge Representation in SQL-TUTOR.
PUB DATE 94
NOTE 7p.; In: Educational Multimedia and Hypermedia, 1994. Proceedings of ED-MEDIA 94--World Conference on Educational Multimedia and Hypermedia (Vancouver, British Columbia, Canada, June 25-30, 1994); see IR 017 359.
PUB TYPE Reports - Descriptive (141) -- Speeches/Conference Papers (150)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS Cognitive Style; Computer System Design; Curriculum Development; Instructional Development; Instructional Materials; *Intelligent Tutoring Systems; *Knowledge Representation; Programming; *Skill Development; Teaching Methods
IDENTIFIERS Computer Architecture

ABSTRACT

Research in the area of knowledge-based tutoring systems (KBTS) and intelligent tutoring systems (ITS) has great implications for computer applications in education and training. In a KBTS, course material is represented independently of teacher procedures, so that problems and remedial comments can be generated differently for each student. Such a system offers instructions in a manner that is sensitive to the student's strengths, weaknesses, and preferred style of learning; in addition, individualized tutoring can be provided. The objective of the research presented in this paper is to develop a framework upon which effective KBTSs can be built. SQL-TUTOR was developed for the domain of SQL programming. In the design of SQL-TUTOR, an architecture based on a layered curriculum is used to organize the system into components. The course curriculum is represented by a curriculum graph; this architecture has such features as explicit curriculum knowledge representation, multiple teaching sequences, multiple viewpoints of a teaching goal, and overall skill training for students. (AEF)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)
☐ This document has been reproduced as received from the person or organization originating it.
☐ Minor changes have been made to improve reproduction quality.
• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
Gary H. Marks
TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC).

ED 388 307

Curriculum Knowledge Representation in SQL-TUTOR

GANG ZHOU
Department of Computer and Information Science
New Jersey Institute of Technology
Newark, NJ 07102

Abstract: The research in the area of Knowledge-Based Tutoring System (KBTS), or Intelligent Tutoring System (ITS), has great importance for computer applications in education and training. The objective of this research is to develop a framework upon which effective KBTSs can be built. We have developed SQL-TUTOR, a KBTS for the domain of SQL programming, which adopts a new architecture called *curriculum oriented* in the sense that both domain knowledge base and student model are built based on the course curriculum. The course curriculum is represented by a *curriculum graph*, which incorporates the knowledge about the goal structure of a course, multiple viewpoints of a goal, and prerequisite relations among different goals. This new architecture has such features as i) explicit curriculum knowledge representation; ii) multiple teaching sequences; iii) multiple viewpoints of a teaching goal; and iv) overall skill training to a student.

Introduction

The goal of KBTS research is to build instructional programs that incorporate well-prepared course material in lessons that are optimized for each student. In a KBTS, course material is represented independently of teaching procedures, so that problems and remedial comments can be generated differently for each student. Such a system can offer instructions in a manner that is sensitive to the student's strength, weakness, preferred style of learning and provide individualized tutoring. In the design of SQL-TUTOR, we have adopted an architecture based on layered curriculum to organize system components[Chan, 1992, Lesgold, 1987]. The idea is to capture a teacher's knowledge of the curriculum at the first place and make this knowledge explicit to the system. We believe that this architecture strongly supports the perspective that views a KBTS as a knowledge communication system[Wenger 1987]. Figure 1 shows the components in our system.

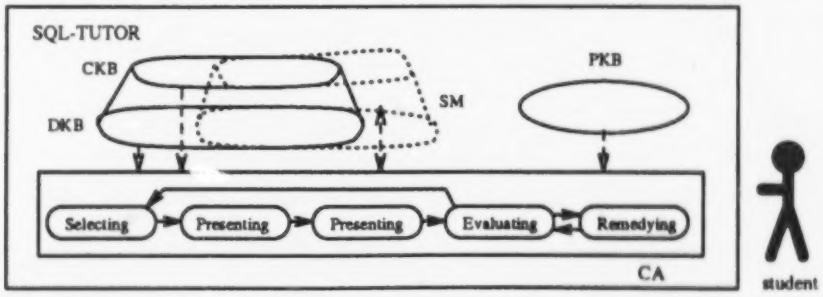

Figure 1: SQL-TUTOR

In this architecture, the knowledge in a KBTS is decomposed into four components and stored in Curriculum Knowledge Base (CKB), Domain Knowledge Base (DKB), Pedagogical Knowledge Base (PKB), and Student Model (SM). Another component in SQL-TUTOR is the Communication Agent (CA), which controls the interactions between SQL-TUTOR and a student. It consists of five modules: i) *selecting module* – selects a new topic for the student to study; ii) *presenting module* – choses the piece of the text to be presented and display it to the student; iii) *answering module* – reads and answers

IR017456

BEST COPY AVAILABLE

the student's questions; vi) *evaluating module* – evaluates the student's performance; and v) *remedying module* – finds a remedical decision and applying it. In the following sections, we will discuss the design of the CKB in detail.

Content Tree and Simple Curriculum Graph

Suppose you are working on editing a textbook. How would you like to organize the subject materials that you are going to teach your students? Most people will organize them by chapters, sections and subsections, and use a contents table to list their titles. This is, in fact, a *decomposing* process. A book is decomposed into chapters, a chapter into sections, and a section into subsections. From a content table, a student can know the structure of the book: chapters, sections, subsections, and their relations. Thus a content table conveys some kind of meta-knowledge, the knowledge about the subject matters. We call this type of knowledge *curriculum knowledge* .

Because it determines the selection and sequencing of topics in a course, a KBTS must formulate a representation of curriculum knowledge. As we switch from the educational perspective that we have just used in our discussion to KBTS perspective, two changes are noteworthy: i) The **subject materials** are the target knowledge that a KBTS tries to impart to a student and is represented by the **domain knowledge** in the system. ii) **decomposing** subject materials is exactly a **subgoaling** process on the domain knowledge, and the resulting *goal-subgoal tree* corresponds to the content table. We call this tree a *content tree* . The goal associated with the root of a content tree is the topic of the book (e.g. teaching SQL). At the next level are a list subgoals, which are the topics of each chapters (e.g. introducing SQL). The goals of each chapter in turn are split into subgoals, which are the topics of each sections (e.g. teaching SQL concepts), and so on. The subgoaling process terminates at some lower level when the domain knowledge has been split into such simple sections that they could be taught to the student completely as one unit. So a leaf of such a tree is also called a *unit*. To summarize, we have the following two definitions.

Definition 1 The knowledge about the domain knowledge in a KBTS is called *curriculum knowledge*.[]

Definition 2 A *content tree* for a course is a tree with the following features: i) each node is associated with a teaching goal (topic); ii) the goal associated with the root is the goal of the course; iii) the goals associated the children of a node is its subgoals (or in another word, the goal at a node consists of a list of subgoals.[]

Figure 2 shows a content table and its corresponding content tree. A node in a content tree is called a *goal node*, or simply a *g-node*. An edge in a content tree is called a *goal edge*, or simply a *g-edge*. A content tree is denoted by $CT=(G, GE, R)$, where G is the set of g-nodes, GE is the set of g-edges, and R is the root of the tree.

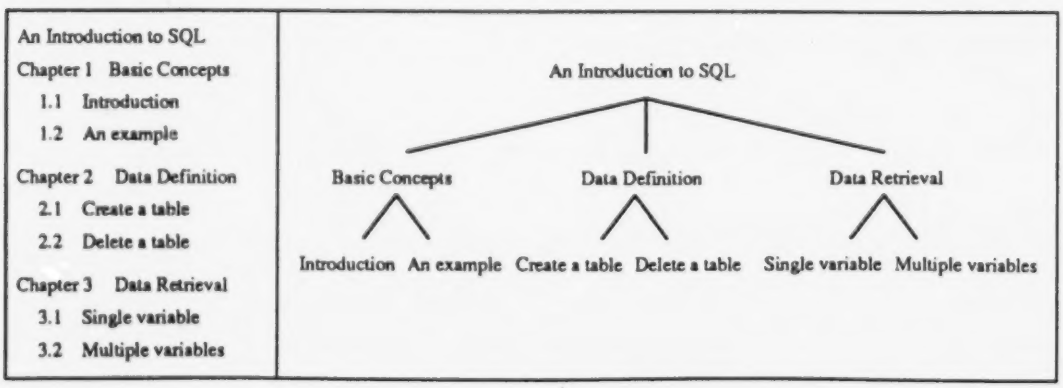

Figure 2: A content table and its corresponding content tree

Although simple, the content tree generated by the subgoaling is just a rough representation of curriculum knowledge in a teacher's mind. It suffers from two major weaknesses:

- *single sequence restriction:* the structure of the course is fixed, and there is only one sequence to go over the whole course. This restriction has the pedagogical meaning that a student can only follow one sequence to study the domain subjects and this sequence contains all the topics of the course. That is, a student has to study all the chapters and all the sections in a chapter in a predefined order to finish the course. Since generally there are several suggested ways to go through those topics of a course (some of them are required for all students, some of them are only appropriate for advanced studies, and some of them are only for readers with special interests), it is more desirable to organize them in such a way that the course could be easily tailored according to individual requirements of each student.

- *single viewpoint restriction:* each teaching goal is represented by only one perspective. This restriction allows the student to study the subject only in one perspective. Because things are not always so simple, a teacher might find that, depending on the different pedagogical purposes, there are different viewpoints on the instruction he wants to present. For example, there are three different viewpoints on the data objects managed by a DBMS: i) from the theory of relational database; ii) from relational algebra; ii) from file management system. Thus we have three sets of terminologies, {*relation, tuple, attribute*}, {*table, row, column*}, {*file, record, field*}, which refer to the same concepts.

In order to the lift first restriction, the knowledge about the *prerequisite* relations between the pieces of the domain knowledge can to added to a content tree.

Definition 3 Let $\mathcal{CT}=(G, GE, R)$ be a content tree and $l_1, l_2 \in G$ be two leaves. Leaf l_2 is said to be *dependent on* leaf l_1, denoted by $\mathbf{dp}(l_1, l_2)$, if the goal at l_1 is a part of the background for teaching the goal at l_2.[]

Definition 4 Let $\mathcal{CT}=(G, GE, R)$ be a content tree and two distinguished nodes n_1, n_2 be siblings. Node n_1 is a *prerequisite* of node n_2, denoted by $\mathbf{pr}(n_1, n_2)$, if either $\mathbf{dp}(n_1, n_2)$, or there exist two leaves l_1 and l_2 such that: i) $\mathbf{pr}(l_1, l_2)$; ii) l_1 is a successor of n_1, and iii) l_2 is a successor of n_2.[]

The prerequisite relation is the most important one among the topics of a course. It can be used to generate or check valid teaching sequences for a course in which a topic can be presented only if all of its prerequisites have been presented. It can also provide useful information to help the system to identify a student's missing conceptions or misconceptions in a diagnosing process. The prerequisite relation has the following properties: i) *irreflexivity*: for all nodes n in a $\mathcal{CT}$, $\mathbf{pr}(n, n)$ is false; ii) *transitivity*: for all nodes n_1, n_2 and n_3 in a $\mathcal{CT}$, if $\mathbf{pr}(n_1, n_2)$ and $\mathbf{pr}(n_2, n_3)$, then $\mathbf{pr}(n_1, n_3)$; In other words, it is a partial relation over the topics of a course.

Definition 5 Let $\mathcal{CG}=(G, GE, R)$ be a content tree and $n_1, n_2 \in G$ be two nodes. $\mathbf{pr}(n_1, n_2)$ is *primitive* if there not exists a node $n_x \in G$ such that both $\mathbf{pr}(n_1, n_x)$ and $\mathbf{pr}(n_x, n_2)$ are true.[]

If we add prerequisite information to a $\mathcal{CT}$, the result is a lattice called *simple curriculum graph*, or $\mathcal{SCG}$. In a simple curriculum graph, there is a directed edge from node n_2 to node n_1 if and only if $\mathbf{pr}(n_1, n_2)$ and it is primitive. Such an edge is celled a *prerequisite edge*, or simply a *p-edge*.

Definition 6 A simple curriculum graph is a four-tuple $\mathcal{SCG}=(G, GE, P, R)$, where G is the set of goal nodes; GE is the set of goal edges; P is the set of prerequisite edges; R is the start node.[]

When selecting a new topic for the student, the system can select any topic as long as the following constraint is satisfied: *if* $\mathbf{pr}(n_1, n_2)$, *then the presenting of* n_2 *must follow the presenting of* n_1. Since this procedure is nondeterministic, the system can generate multiple teaching sequences.

Curriculum Graph

Sometimes, there are several valid ways to decomposed a goal into a list of subgoals. The different decompositions view the same goal from different perspectives. So, from any specific point of view, there are clear pathways that determine the sequencing of instruction, though of course there are other alternative sequences to approach the same goal. There may be psychological emotions which result in individual differences in aptitude or preference for different viewpoints.

As an example of multiple effective perspectives to view a teaching goal, consider the decomposition of SELECT clause of SELECT statement of SQL language. We can partition this goal according to

the types of the data objects they applied to, resulting in two subgoals **SINGLE-COLUMN-RETRIEVAL, MULTIPLE-COLUMN-RETRIEVAL**; or partition them into those that have or have no qualifiers, resulting in **UNQUALIFIED-RETRIEVAL, QUALIFIED-RETRIEVAL**. If we incorporate this feature of multiple viewpoints to a simple curriculum graph, the result is a *curriculum graph*, or simply $\mathcal{CG}$. In such a graph, the children of a node may represent the multiple viewpoints about the teaching goal of the node. We call this type of node a *multiple viewpoints node*, or simply a *m-node*, and an edge connecting a *m-node* and its children a *multiple viewpoints edge*, or simply a *m-edge*.

Definition 7 A *Curriculum Graph* is a six-tuple $\mathcal{CG}=(G, M, GE, ME, P, R)$, where G is the set of goal nodes; M is the set of multiple viewpoints nodes; GE is the set of goal edges; ME is the set of multiple viewpoint edges; P is the set of prerequisite edges; R is the start node.□

Figure 3 is a curriculum graph evoluted from the content tree shown in Figure 2, where **Data Retrieval** a m-node with two viewpoints: based on data objects (DBR) and based on qualification (QBR). Both of them are g-nodes.

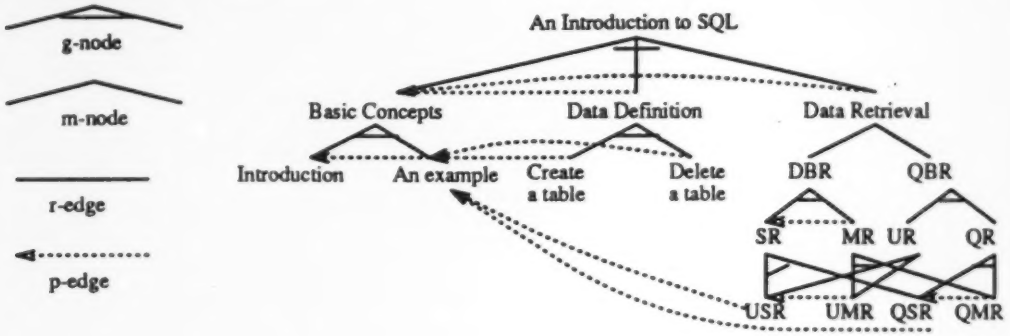

Figure 3: A Curriculum Graph

A r-node or m-node is also called a *regular nodes*, or simply *r-nodes*. A g-edge or m-edge is called a *regular edge*, or simply a *r-edge*. A node m is a *regular ancestor (successor)* of another node n in a $\mathcal{CG}$ if they are connected by a series of regular edges. In the curriculum graph of Figure 3, the regular ancestor of **SCUS** include these node: **An Introduction of SQL, Data Retrieval, RBOD, RBOQ, SCR, UR**. What is noteworthy is that the decompositions of the multiple viewpoints of a m-node should always produce the same set of nodes at some lower level. We call this set of nodes the *base* of the goal. As shown in Figure 3, the base of **Data Retrieval** consists of four nodes **SCUR** (Single Column Unqualified retrieval), **MCUR** (Multiple Column Unqualified retrieval), **SCQR** (Single Column Qualified Retrieval), and **MCQR** (Multiple Column Qualified Retrieval). Nodes in a base are not necessarily units. Some or all of them can be interior nodes. The following is the formal definition of a base.

Definition 8 Let $\mathcal{CG}=(G, M, GE, ME, P, R)$ be a curriculum graph, $m \in M$ be a m-node and $n_1, \ldots, n_k \in (G \cup M)$ be its children. The *base* of m, denoted by $base(m)$, is the set of nodes with all of $n_1, \ldots, n_k$ as their only regular ancestors.□

The requirement that all the children of a m-node map into the same base at some lower level in the curriculum graph implies two things: i) the child of a m-node can not be a leaf; ii) all the nodes in $base(m)$ have k direct regular parents, if m has k children.

Study Graph and its Properties

Definition 9 Let $\mathcal{CG}=(G, M, GE, ME, P, R)$ be a curriculum graph, and $N_s = \{n_1, n_2, \ldots, n_k\}$ be a set of leaves. A *study graph* of N_s is a connected subgraph of $\mathcal{CG}$ with start node R and tipnodes $n_1, n_2, \ldots, n_k$. It is *well defined* if for any nodes n_i, n_j, if $\mathbf{pr}(n_i, n_j)$ and $n_j \in N_s$, then $n_i \in N_s$.□

Generally speaking, for a set of given nodes N_s, there are several study graphs, and we denote them by $\mathbf{sg}(\mathcal{CG}, N_s)$. Obviously, the curriculum graph itself is the maximum well defined study graph of its leaves.

Definition 10 Let $\mathcal{CG}$=(G, M, GE, ME, P, R) be a curriculum graph and $l \in G$ be a leaf. The *a-closure* of l in $\mathcal{CG}$, denoted by $\mathbf{a}^*(\mathcal{CG}, l)$, is the set of prerequisite nodes of l, or $\mathbf{a}^*(\mathcal{CG},l) = \{m|\mathbf{pr}(m,l), m \in G\}$. The *s-closure* of l in $\mathcal{CG}$, denoted by $\mathbf{s}^*(\mathcal{CG},l)$, is the set of nodes which take n as their prerequisite, or $\mathbf{s}^*(\mathcal{CG},l) = \{m|\mathbf{pr}(l,m), m \in G\}$[]

Definition 11 Let $\mathcal{CG}$=(G, M, GE, ME, P, R) be a curriculum graph and $n \in G$ be a node. The *learning graph* of n with respect to $\mathcal{CG}$, denoted by $\mathbf{lg}(\mathcal{CG},n)$, is defined as: i) if $n \in G$ is a leaf, then its *learning graph* is the *maximum* study graph of its a-closure, or $\mathbf{lg}(\mathcal{CG}, n) = \mathbf{sg}(\mathcal{CG}, \mathbf{a}^*(\mathcal{CG},n))$; ii) if $n \in (G \cup M)$ is not a leaf, then its *learning graph* is the *maximum* study graph of the union of the a-closures of those leaves which are the regular successors of n, or $\mathbf{lg}(\mathcal{CG}, n) = \mathbf{sg}(\mathcal{CG} \cup \mathbf{a}^*(\mathcal{CG},l))$, where, l is a leaf and regular successor of n.[]

In Figure 4, part (a) and (b) are two study graphs for {UMR} and {UMR, QSR} of the curriculum graph shown in Figure 3. Part (c) and (d) are the learning graphs of leaf QMR and interior node SR, respectively.

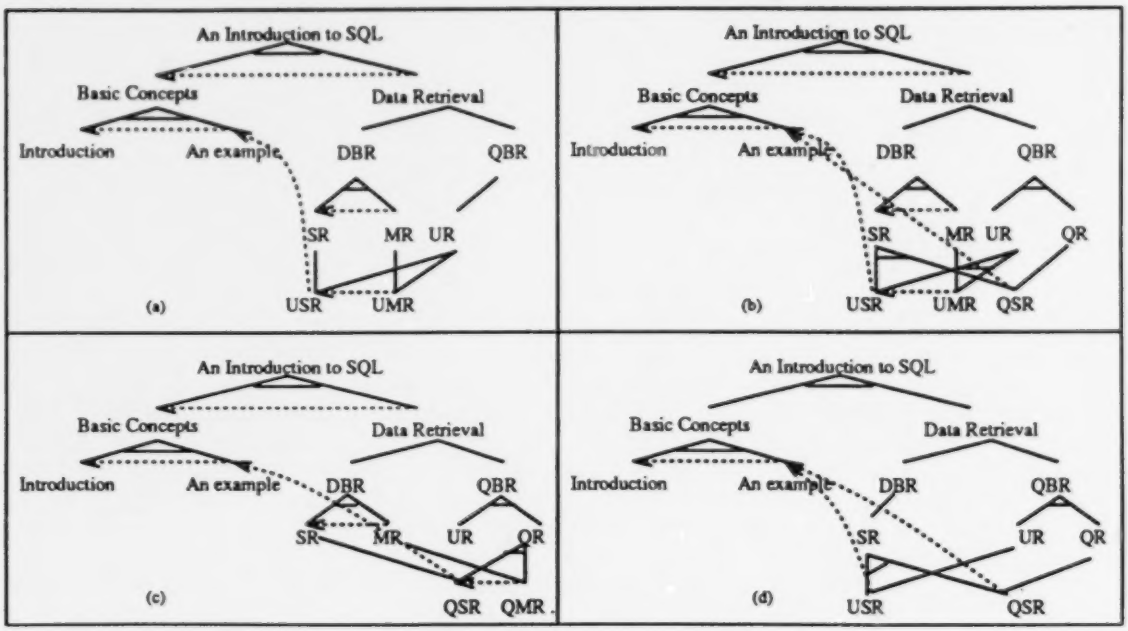

Figure 4: Examples of study graph and learning graph

Theorem 1 *A learning graph is always a well defined study graph.*[]

PROOF This is obvious from the definition, because for any leaf of a learning graph, all of its prerequisites are also there.

Given a curriculum graph $\mathcal{CG}$ and a m-node of $\mathcal{CG}$, if we only focus on one viewpoint and discard all others, do we still have a well defined curriculum? For the same graph, if we remove some node from it, how can we keep the remaining part well defined? The following theorems will answer these questions.

Theorem 2 *If $\mathcal{CG}$=(G, M, GE, ME, P, R) is a curriculum graph, $m \in M$ is a m-node, and $n_i \in (M \cup G)$ is a child of m, then if we remove all the nodes which are regular successors of m, but not regular successors of n_i, from $\mathcal{CG}$, the result is still a well defined study graph.*[]

PROOF Omitted.

Theorem 3 *If $\mathcal{CG}$=(G, M, GE, ME, P, R) is a curriculum graph, L is the set of its leaves, and $n \in L$ is a leaf, then $\mathbf{sg}(\mathcal{CG}, L-(\{n\} \cup \mathbf{s}^*(\mathcal{CG},n)))$ is a set of well defined study graphs.*[]

PROOF Omitted.

Corollary 1 *If $\mathcal{CG}$=(G, M, GE, ME, P, R) is a curriculum graph, L is the set of leaves, and $N_s = \{n_1, n_2, \ldots, n_k\} \subset L$ is a subset of leaves, then $\mathbf{sg}(\mathcal{CG}, L-(N_s \cup \mathbf{s}^*(\mathcal{CG},n_1) \cup \ldots \cup \mathbf{s}^*(\mathcal{CG},n_k)))$ is a set of well defined study graphs.*

PROOF Omitted.

Definition 12 Let $CG=(G, M, GE, ME, P, R)$ be a curriculum graph and $n \in (G \cup M)$ be a node. n is *monous* if all of its regular ancestors are g-nodes. n is *single parent* if it only has one regular incoming edge. []

Theorem 4 *If $CG=(G, M, GE, ME, P, R)$ is a curriculum graph and $n \in (G \cup M)$ is a non-monous interior node, then the removal of n and all its single parent successors from CG results in a well defined study graph.*[]

PROOF Omitted.

An Object-Oriented Implementation

We have implemented a curriculum management system called **CM** (Curriculum Manager) by Common Lisp Object System (CLOS)[Keene, 1989]. Four types of classes have been defined in **CM**: **node**: the foundation of all nodes in a curriculum graph; **goal-node**: whose instances are g-nodes; **multi-view-node**: whose instances are m-nodes; **unit**: whose instances are units.

The system interface to a student is defined by a group of system commands: **display, focus, study, not-study, and skip**. These commands provide tools for users to construct course curriculum according to their individual requirements before the selecting period. The purpose of **focus** is to allow a student to focus on one view point while ignoring others on a goal. A student can use **study** to a select the specific topic he wants to study. The result is a new curriculum graph (study graph) with all the prerequisite nodes remained in positions but unrelated topics removed. On the other hand, he can use **not-study** to exclude whatever optional topic he is not interested in, while keeping the remaining curriculum well defined (in the sense that the result is a well defined study graph). The last operator is **skip**, is used to allow students to skip any part of a course. It differs from **not-study** in that the resulting study graph is not necessarily well defined. When students do not want to study some topic (because they have already mastered it), they can use **skip** to skip it; on the other hand, even if the students have not mastered some topic, they still can get rid of it by **not-study** (one possible reason is they have no interest in it).

In comparison with the existing KBTSs, our system has the following three major advantages:

- The curriculum knowledge about a course is represented explicitly in the system and the CKB is built up based on a theory of curriculum knowledge representation;
- The structure of a course could be tailored and multiple curriculum could be generated according to the special interests from students, thus each student can chose an individualized course from one system;
- Student can study the same part of the domain knowledge from different conceptual perspectives, and combined with the above feature, this offers well-prepared course material in lessons that are optimized for each student.

References

[Chan, 1992] Chan, Tak-Wai. (1992). Curriculum Tree: A Knowledge-Based Architecture for Intelligent Tutoring Systems. *Intelligent Tutoring Systems: Second International Conference on Intelligent Tutoring Systems, ITS'92*. Spring-Verlag.

[Lesgold, 1987] Lesgold, A. (1987). Toward a Theory of Curriculum for Use in Designing Intelligent Instructional Systems. *Learning Issues for Intelligent Tutoring Systems*. Springer-Verlag, New York.

[Keene, 1989] Keene, S.E. (1989) *Object-oriented Programming in Common LISP: a programmer's guide to CLOS*. Addison-Wesley.

[Wenger 1987] Wenger, E. (1987) *Artificial Intelligence and Tutoring Systems: Computational and Cognitive Approaches to the Communication of Knowledge*. Los Altos, California: Morgan Kaufmann Publishers, Inc.